**RC23961 (W0605-108) May 18, 2006**
**Computer Science**

# IBM Research Report

## Obtaining Formal Knowledge from Informal Text Analysis

**J. William Murdock, Christopher Welty**
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

# Obtaining Formal Knowledge from Informal Text Analysis

J. William Murdock & Christopher Welty
IBM Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532, USA
{murdockj, welty}@us.ibm.com

## Abstract

Populating formal knowledge-bases from natural-language text is a long-standing objective in computer science. Recent advancements in both ontology research and information extraction research are making this objective increasingly obtainable. However, there are still serious obstacles to performing automated reasoning over the contents of text documents. This paper focuses on one of those obstacles: differences between the formal ontologies used by reasoning systems and the informal ontologies used by extraction systems. We describe a framework for automating translation from extracted information to formal knowledge, and we describe a complex, implemented system that uses this framework. We also describe results from this system applied to a moderately large (approximately 75 MB) text corpus.

## 1. Introduction

Ontologies describe the kinds of phenomena (e.g., people, places, events, relationships, etc.) that can exist. Reasoning systems typically rely on ontologies that provide extensive formal semantics, since those semantics enable those systems to draw complex conclusions. In contrast, systems that extract information from text typically use much lighter-weight ontologies to encode their results, because those systems are generally *not* designed to enable complex reasoning.

The Unstructured Information Management Architecture, UIMA, (Ferrucci & Lally, 2004) is an open-source middleware platform for integrating components that analyze unstructured sources such as text documents. UIMA-based systems define “type systems” (i.e., ontologies with extremely limited semantic commitments) to specify the kinds of information that they manipulate (Götz & Suhre, 2004). UIMA type systems include a type/subtype hierarchy (single inheritance only) and allow atomic domain/range restrictions on properties (i.e., one can specify a “feature” that connects instances of one specified type to instance of another specified type). These capabilities enable very few inferences (e.g., if A is an instance of a type X, and the supertype of X is Y, then A is an instance of Y). Thus to do substantive reasoning over the results of UIMA-based extraction, one needs to convert results into a more expressive ontology than the UIMA type system language allows.

## 2. Related Work

The work reported in this paper bridges a gap between two existing lines of research:

- Research on extraction of formal knowledge from text (e.g., Dill, Eiron, et al. 2003) typically assumes that text analytics are written for the ontology that the knowledge should be encoded in. Building extraction directly on formal ontologies is particularly valuable when the extraction is intended to construct or modify the original ontology (Maynard, Yankova, et al. 2005; Cimiano & Völker, 2005). However, there is a substantial cost to requiring text analytics to be consistent with formal ontology languages. There are many existing systems that extract entities and relations from text using informal ontologies that make minimal semantic commitments (e.g., Marsh, 1998; Byrd & Ravin, 1999; Liddy, 2000; Miller, Bratus, et al., 2001; Doddington, Mitchell, et al., 2004). These systems use these informal ontologies because those ontologies are relatively consistent with the way concepts are expressed in text and are well-suited for their intended applications (e.g., document search, content browsing). However, those ontologies are not well-suited to applications that require complex inference. Our work provides an explicit knowledge integration step that allows us to populate semantically-rich ontologies using content that was extracted by extraction systems that provide semantically-impoverished outputs.

- Research on schema matching for knowledge integration (e.g., Milo & Zohar, 1998; Noy & Musen, 2001) typically focuses on finding very simple (e.g., one-to-one) mappings among terms in ontologies. Schema matching is useful when the ontologies are large and complex, so that these mappings, while individually simple, are numerous and challenging to find. Our work focuses on the opposite circumstance: We assume that the ontologies are small and manageable enough that one can find the correspondences manually and that the mappings may be more complex (conditional, many-to-many, etc.) than an automated matching system can handle.

The difference between our work and schema matching work is largely driven by our experience with the problem of extracting formal knowledge from text. Adding a term to a text extraction ontology is generally expensive, since one needs to supply training data and/or patterns for recognizing that term. Consequently, text extraction ontologies rarely have more than a few dozen terms, and we are not aware of any with more than a few hundred. Thus manually finding which terms in an extraction ontology map to terms in some other ontology generally involves a fairly small amount of effort.

Schema-matching technologies have typically been used when the applications that the source and target ontologies were designed for are identical or at least quite similar; e.g., matching one e-commerce database schema to another, matching one theorem-proving ontology to another. In those cases, the assumption that individual mappings will tend to be very simple can be valid; since the designers of the ontologies had the same basic purpose in mind, it is plausible that much of their content will involve slightly different terms that mean the same thing. Mapping extracted information into formal reasoning ontologies does not have this characteristic; these applications are radically different and tend to lead to radically different conceptualizations of basic content. The mapping of temporal information discussed in Section 4 illustrates this trend. For these sorts of differences, it is not feasible to restrict the mappings between terms to be sufficiently simple and obvious enough that they can be discovered by state-of-the-art fully-automated matching techniques.

## 3. KITE Architecture

KITE is a middleware platform for use by developers of knowledge integration applications. KITE consists of two major components:

- **KITE Core Framework**: Java interfaces, data structures, and a central control mechanism for mapping entities and relationships from one ontology to another.
- **KITE Commons**: A set of broadly applicable plugins that comply with the interfaces specified in the core framework.

Figure 1 shows an abstract view of the architecture of a knowledge-integration system implemented on KITE. A KITE-based integrator takes as input a *Source Repository* (e.g., a database, an RDF XML file). Information in that repository is encoded in the *Source Ontology* (which is accessed via an *Ontology Language Plugin*). The Source Plugin reads from the source repository and outputs *Source Data* encoded in KITE data structures for instances and tuples. *Mapper Plugins* may be primitive or aggregate. Aggregate mapper plugins are composed of other (primitive or aggregate) mapper plugins. Primitive mapper plugins are Java objects that take *Source Data* as input and output *Target Data* (which consist of the same data structures, but are encoded in the *Target Ontology*). The *Target Plugin* writes that data to a *Target Repository* and the *Provenance Plugin* writes the mappings from source to target data into a *Provenance Repository*.

Some knowledge integration systems can be built using only the KITE Core Framework plus plugins in the KITE Commons (i.e., the plugins included with KITE). For example, the KITE Commons includes source, target, and ontology language plugins for OWL and for UIMA type systems, and KITE provides a mapper that performs one-to-one mappings using an XML lookup table. Thus a developer who wants to write an integrator that maps UIMA extracted information into an OWL ontology that has a one-to-one correspondence with the extraction types can simply combine these existing plugins and provide an XML file listing the one-to-one mappings.

However, in our experience, most integration problems cannot be solved using *only* the

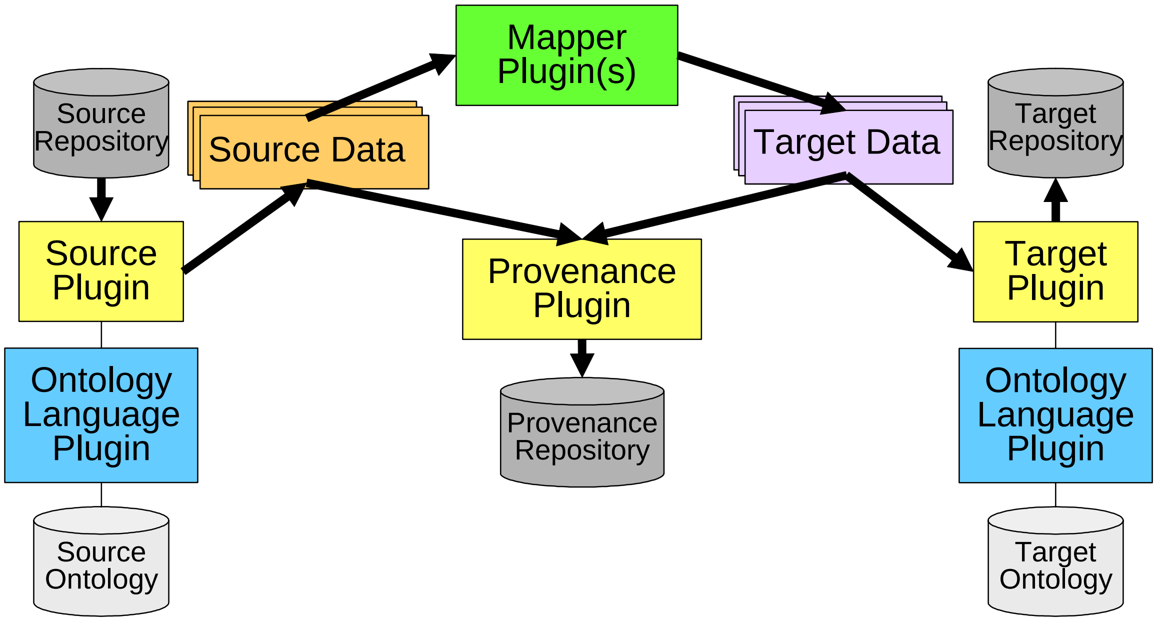

**Figure 1**: Abstract view of a KITE-based knowledge integrator.

KITE Commons plugins. For those problems, developers need to supply their own plugins for some portions of the integration process. For example, some types in the source ontology may have a complex m-to-n mapping to the target ontology; a developer would need to create specialized mapping software written or wrapped in Java to handle those mappings. If these ontologies also had other types that did map one-to-one, then those types could be handled by KITE's table mapper. The integrator developer would specify an aggregate mapper that includes both the custom-built primitive mapper and KITE's table mapper. This aggregate mapper could then be used with KITE Commons plugins for repositories, ontologies, etc. (if the system uses repository formats, ontology languages, etc. that the KITE Commons plugins handle) or with custom-built plugins (for other formats, etc.).

## 4. Example Knowledge Integrator

Figure 2 shows an example of a KITE-based knowledge integrator. Source data for this application is encoded in HUTT (Hierarchical Unified Type Taxonomy), a UIMA type system based on a variety of established information extraction taxonomies (e.g., Doddington, Mitchell, et al., 2004; Sauri, Litman, et al., 2004). The output ontology for this application is the OWL ontology used in the KANI project (Fikes, Ferrucci, & Thurman, 2005).

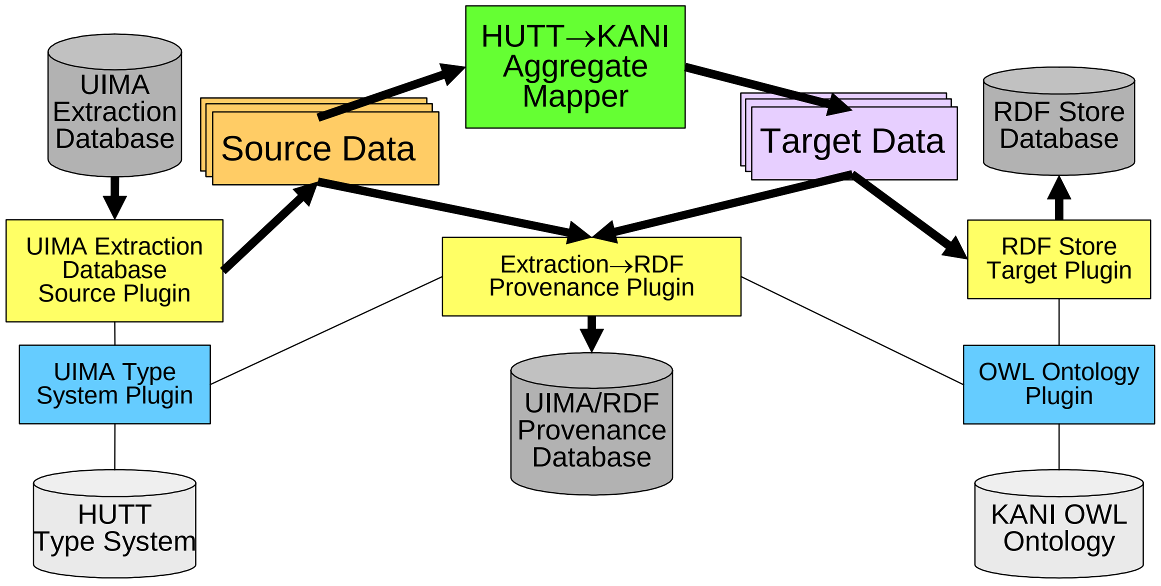

**Figure 2**: Example KITE-based application

This input data for the example application is stored in a relational database designed to contain UIMA extracted information. The KITE Commons includes a plugin (*UIMA Extraction Database Source Plugin*) that accesses this database and outputs KITE instances and tuples (*Source Data*). This source data is provided to an aggregate mapper composed of an assortment of both generic mappers from the KITE Commons and specialized mappers that were written for the HUTT to KANI integrator. These mappers output target data. That data is consumed by two plugins from the KITE Commons: the

*RDF Store Target Plugin* writes the target data alone into a relational database for RDF triples, and the *Extraction → RDF Provenance Plugin* records (potentially complex) mappings from source data in the extraction database to target data in the RDF database; these mappings are stored in the *UIMA/RDF Provenance Database*.[1]

Systems that access instances and triples from the RDF store can request traces of the information extraction and knowledge integration processes that created those instances and triples. The provenance database is able to return that information either as database entries or in the OWL-based Proof Markup Language, PML (Pinheiro da Silva, McGuinness & Fikes, 2006). Systems that perform additional reasoning over the extracted knowledge can provide integrated end-to-end PML traces that explain their conclusions as a combination of logical inferences from the RDF knowledge and extraction inferences used to obtain that knowledge from text (Welty, Murdock, et al., 2005).

As mentioned above, the aggregate mapper used in this application contains a combination of mappers from the KITE Commons and mappers that were written for this application (specifically, it includes five mapper plugins from the KITE Commons and four custom mapper plugins). The most complex mappers that were written for this application involve the handling of temporal information. The representation of time in HUTT is based on TimeML (Sauri & Littman, 2004), a language for marking up expressions of time in natural-language text. The representation of time in the KANI ontology is OWL-Time (Hobbs, 2004), a semantic web ontology. OWL-Time makes relatively subtle distinctions that are usually implicit in text (e.g., distinguishing between time intervals and time interval descriptions). Furthermore, OWL-Time has distinct properties to encode different aspects of a description of a time (year, month, day, hour, etc.). In contrast, TimeML does not encode a time and its expression separately, and uses a relatively compact normalized form to encode a full time description in a single string. These differences are motivated by the different applications that these ontologies were designed for; OWL-Time directly enables a wide variety of logical inferences about times, while TimeML provides a convenient and compact formalism for identifying, normalizing, and linking expressions of time in text.

The relationship between TimeML and OWL-Time illustrates our motivation for designing KITE as a plugin framework that allows users to develop their own mappers and combine them into aggregates. A generic mapping component that was expressive enough to handle the mapping between these two portions of the HUTT and KANI ontologies would be extremely complicated to develop and to use. However, many of the other terms in HUTT and KANI *are* handled easily by simple, generic mappers. For this application, KITE has enabled an effective combination of generic and special-purpose mapping capabilities.

## 5. Status

The example application discussed in the previous section has been used on a variety of text corpora to enable reasoning, knowledge-based search, and browsing of knowledge

[1] The extraction, RDF, and provenance databases used in this example system are all part of a larger UIMA data storage service called EKDB: Extracted Knowledge Database (IBM Research, 2005).

and provenance. The largest corpus we have run this application on is approximately 75 MB of text; from this text, we extracted approximately 2 million RDF triples. The various text extraction and indexing processes took about 38 hours for this corpus, and the KITE-based knowledge integrator took about 2 hours to convert the results from the HUTT type system to the KANI ontology.

Examples of other systems that have been built using KITE include:

- A knowledge integrator that translates semantic search queries (Brown, Dolbey, & Hunter, 2003) from a relatively "user friendly" OWL ontology into HUTT, the UIMA type system discussed in Section 4. This knowledge integrator allows users to specify their queries in the OWL ontology and have them addressed by a fast, scalable HUTT-based semantic search index. We have built HUTT-based semantic search indexes for corpora up to 3 GB in size. Semantic search is an application that does not require complex reasoning, so it is not necessary to translate the extracted information from the text into a formal ontology for this purpose. Translating queries allows applications to provide user interaction in terms of a formal ontology without having to convert stored data into that ontology.
- A knowledge integrator that generates an OWL ontology (classes and properties) from entity and relation types in a UIMA type system. This system treats each type as a separate instance in the source data and encodes UIMA's type/subtype hierarchy as tuples that relate those types.

KITE is currently in undergoing internal alpha-testing within IBM Research. We expect to make this component available to external users as an internet download when the core API's have become sufficiently stable and hardened. We expect future versions of the KITE platform will include tool support to make development and integration of KITE-compliant components and systems easier. We may develop some of these tools ourselves; however, we expect to make other tools available by more closely aligning KITE with existing standards for which extensive tool support already exists, e.g., UIMA and EMF (Budinsky, Steinberg, et al., 2004).

## 6. Conclusions

There are many existing systems that extract entities and relations from text, and there are many reasoning applications that require entities and relations. However, existing work combining these capabilities has met with only limited success. One reason why such combinations are so challenging is the dramatic differences between formal reasoning ontologies and information extraction ontologies. The KITE middleware platform provides a foundation for bridging this gap. We have demonstrated that KITE can integrate radically different ontologies at moderately large scales (millions of triples). The use of KITE in several distinct applications provides preliminary evidence of KITE's flexibility. As KITE's user base expands and the framework matures, we expect it to provide an increasingly powerful foundation for populating formal ontologies from unstructured data.

## Citations

Eric W. Brown, Andrew Dolbey, & Lawrence Hunter. 2003. IBM Research and the University of Colorado - TREC 2003 Genomics Track. *12th Twelfth Text REtrieval Conference*.

Frank Budinsky, Ray Ellersick, Timothy J. Grose, Ed Merks, & David Steinberg. 2004. *Eclipse Modeling Framework*. Addison-Wesley.

Roy Byrd & Yael Ravin. 1999. Identifying and Extracting Relations in Text. *4th International Conference on Applications of Natural Language to Information Systems* (NLDB). Klagenfurt, Austria.

Philipp Cimiano, Johanna Völker. 2005. Text2Onto - A Framework for Ontology Learning and Data-driven Change Discovery. *10th International Conference on Applications of Natural Language to Information Systems* (NLDB). Alicante, Spain.

Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, R. Guha, Anant Jhingran, Tapas Kanungo, Sridhar Rajagopalan, Andrew Tomkins, John A. Tomlin, & Jason Y. Zien. 2003. SemTag and Seeker: Bootstrapping the semantic web via automated semantic annotation. *12th International World Wide Web Conference* (WWW), Budapest, Hungary.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, & Ralph Weischedel. 2004. Automatic Content Extraction (ACE) program - task definitions and performance measures. *Fourth International Conference on Language Resources and Evaluation* (LREC).

David Ferrucci & Adam Lally. 2004. UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering* 10 (3/4): 327–348.

Richard Fikes, David Ferrucci, & David Thurman. 2005. Knowledge Associates for Novel Intelligence (KANI). *2005 International Conference on Intelligence Analysis* McClean, VA.

T. Götz & O. Suhre. 2004. Design and implementation of the UIMA Common Analysis System. *IBM Systems Journal* 43 (3): 476-489.

Jerry R. Hobbs. 2004. An OWL Ontology of Time. http://www.isi.edu/~pan/time/owl-time-july04.txt

IBM Research. 2005. Component Services for Knowledge Integration in UIMA (a.k.a. SUKI). http://www.research.ibm.com/UIMA/UIMA%20Knowledge%20Integration%20Services.pdf

Elizabeth D. Liddy. 2000. Text Mining. *Bulletin of American Society for Information Science & Technology*.

Elaine Marsh. 1998. TIPSTER information extraction evaluation: the MUC-7 workshop.

Diana Maynard, Milena Yankova, Alexandros Kourakis, and Antonis Kokossis. 2005. Ontology-based information extraction for market monitoring and technology watch. ESWC Workshop “End User Apects of the Semantic Web,” Heraklion, Crete, May, 2005.

Scott Miller, Sergey Bratus, Lance Ramshaw, Ralph Weischedel, Alex Zamanian. 2001. FactBrowser demonstration. *First international conference on Human language technology research* HLT '01.

T. Milo, S. Zohar. 1998. Using Schema Matching to Simplify Heterogeneous Data Translation. VLDB 98, August 1998.

N. F. Noy & M. A. Musen. 2001. Anchor-PROMPT: Using Non-Local Context for Semantic Matching. *Workshop on Ontologies and Information Sharing at the Seventeenth International Joint Conference on Artificial Intelligence* (IJCAI-2001), Seattle, WA.

Roser Sauri, Jessica Littman, Robert Gaizauskas, Andrea Setzer, & James Pustejovsky. 2004. TimeML Annotation Guidelines, Version 1.1. http://www.cs.brandeis.edu/%7Ejamesp/arda/time/timeMLdocs/guidetest.pdf

Paulo Pinheiro da Silva, Deborah L. McGuinness & Richard Fikes. A proof markup language for Semantic Web services. 2006. *Information Systems* 31(4-5): 381-395.

Christopher Welty, J. William Murdock, Paulo Pinheiro da Silva, Deborah L. McGuinness, David Ferrucci, Richard Fikes. 2005. Tracking Information Extraction from Intelligence Documents. *2005 International Conference on Intelligence Analysis*, McLean, VA.